Processing Data Streams: An (Incomplete) Tutorial

Johannes Gehrke
Department of Computer Science
johannes@cs.cornell.edu
http://www.cs.cornell.edu

Standard Pub/Sub

- Publish/subscribe (pub/sub) is a powerful paradigm
  - Publishers generate data
    - Events, publications
  - Subscribers describe interests in publications
    - Queries, subscriptions
- Asynchronous communication
  - Decoupling of publishers and subscribers
- Much commercial software …

Cornell University

Limitation of Standard Pub/Sub

- Scalable implementations have very simple query languages
  - Simple predicates, comparing message attributes to constants
  - E.g., topic='politics' AND author='J. Doe'
- Individual events vs. event sequences
- Many *monitoring applications* need sequence patterns
  - Stock tickers, RSS feeds, network monitoring, sensor data monitoring, fraud detection, etc.

Cornell University

Example: RSS Feed Monitoring

- Once CNN.com posts an article on Technology, send me the first post referencing (i.e., containing a link to) this article from the blogs to which I subscribe
- Send postings from all blogs to which I subscribe, in which the first posting is a reference to a sensitive site XYZ, and each later posting is a reference to the previous.

Cornell University

Example: System Event Log Monitoring

- In the past 60 seconds, has the number of failed logins (security logs) increased by more than 5? (break-in attempt)
- Have there been any failed connections in the past 15 minutes? If yes, is the rate increasing?
- Have there been any disk errors in the past 30 minutes? If yes, is the rate increasing? (failed disk indicator)
- Have there been any critical errors (those added to the dbase table to monitor by administrators) in the past 10 minutes?

Cornell University

Example: Stock Monitoring

- Notify me when the price of IBM is above $83, and the *first* MSFT price afterwards is below $27.
- Notify me when some stock goes up by at least 5% from one transaction to the next.
- Notify me when the price of any stock increases monotonically for ≥30 min.
- Notify me when the next IBM stock is above its 52-week average.

Cornell University

Solutions?

- Traditional pub/sub
  - Scalable, but not expressive enough
- Database Management System
  - Static datasets
  - One-shot queries
  - Triggers
- Data Stream Management Systems
- Event Processing Systems

Cornell University

Real-Time DSP Requirements

(1) Support a high-level "StreamSQL" language
(2) Deal with out-of-order data
(3) Generate predictable and repeatable outcomes
(4) Integrate well with static data
(5) Fault-tolerance
(6) Scale with hardware resources
(7) Low latency → process data as it streams by ("in-stream processing"); no requirement to store data first

Cornell University

Comparison of Stream Systems

		Number of concurrent queries	
		Few	Many
Complexity of queries	Low	☺	Publish/ subscribe
	High	DSMS	CEP

Cornell University

Tutorial Outline

- Basics
- How to model time
- Data stream query languages and processing models
- Fault tolerance
- New operators
- A Case Study

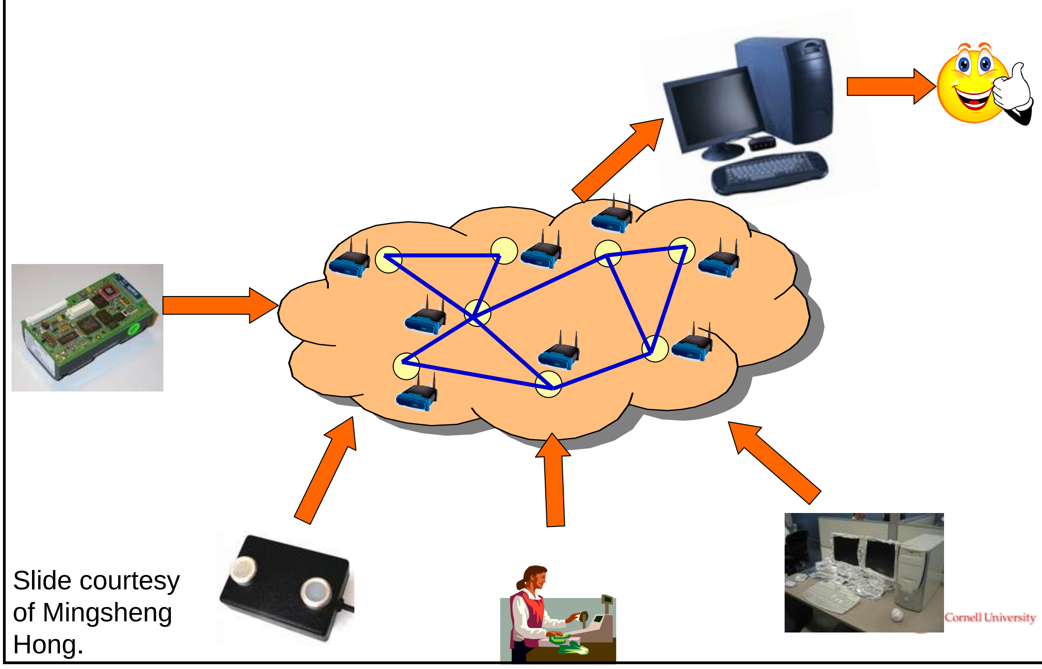
Cornell University

Temporal Model

- Questions:
  - How are timestamps defined?
  - What is the timestamp of an output record?
- Approaches:
  - Point timestamps
  - Interval timestamps
- Surprises like E1;(E2;E3)=E2;(E1;E3)?

Cornell University

Imperfections in Event Streaming

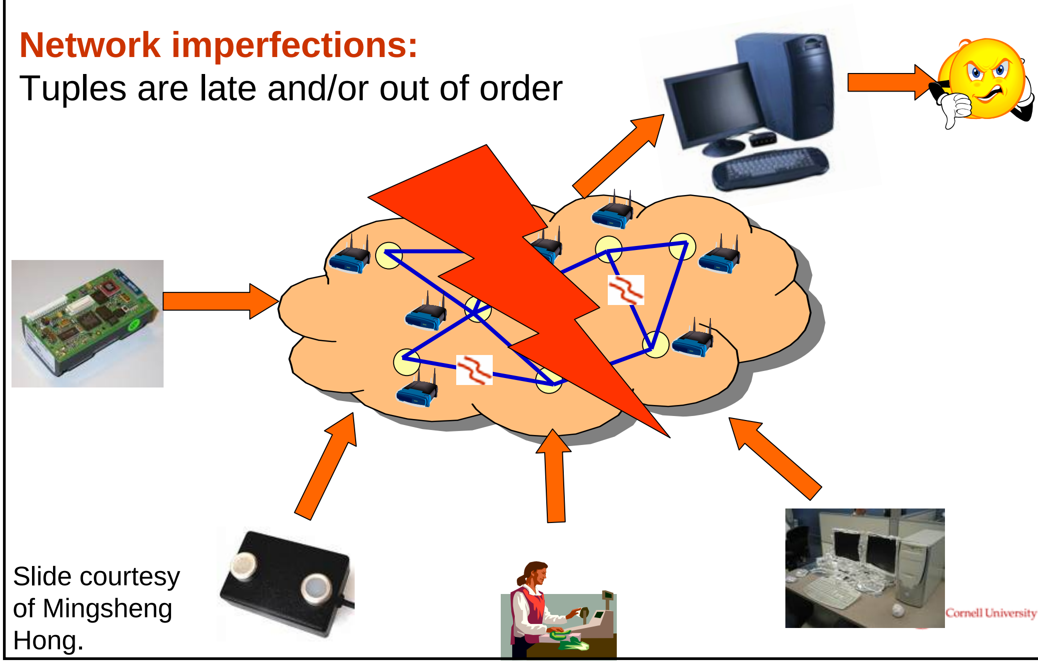


Slide courtesy of Mingsheng Hong.

Cornell University

Imperfections in Event Streaming

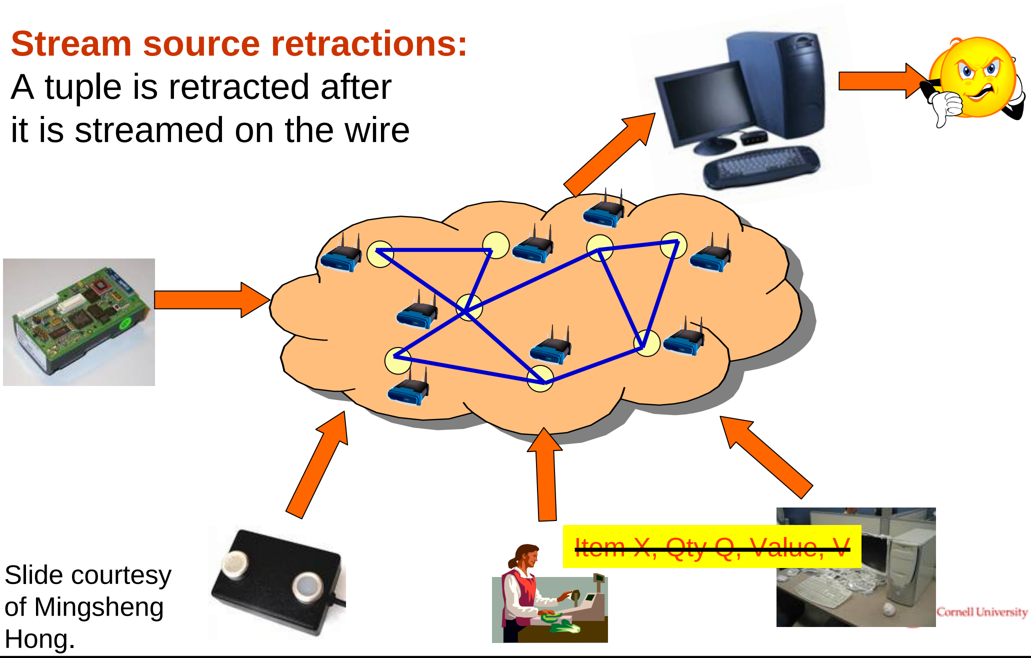
Network imperfections:
Tuples are late and/or out of order



Slide courtesy of Mingsheng Hong.

Imperfections in Event Streaming

Stream source retractions:
A tuple is retracted after it is streamed on the wire



Slide courtesy of Mingsheng Hong.

Data Model

- Stream *S* is a sequence of tuples $\langle \overline{a}; t_0, t_1 \rangle$
  - $\overline{a} = (a_1, \ldots, a_n)$ are data attribute values
    - Like relational tuples
  - *t*'s are temporal values
    - *Starting* and *detection* times of an event
    - Events have duration
  - Example
    - Schema of stock ticker stream: (Name, Price)
    - Base stream events: (IBM, 85; 9:15, 9:15), (MSFT, 27; 9:16, 9:16), (DELL, 29; 9:17, 9:17)

Data Model

- Stream *S* is a sequence of tuples $\langle \boxed{\overline{a};} t_0, t_1 \rangle$
  - $\boxed{\overline{a} = (a_1, \ldots, a_n)}$ are data attribute values
    - Like relational tuples
  - *t*'s are temporal values
    - *Starting* and *detection* times of an event
    - Events have duration
  - Example
    - Schema of stock ticker stream: (Name, Price)
    - Base stream events: (IBM, 85; 9:15, 9:15), (MSFT, 27; 9:16, 9:16), (DELL, 29; 9:17, 9:17)

Data Model

- Stream *S* is a sequence of tuples $\langle \overline{a}; \boxed{t_0, t_1} \rangle$
  - $\overline{a} = (a_1, \ldots, a_n)$ are data attribute values
    - Like relational tuples
  - *t*'s are temporal values
    - *Starting* and *detection* times of an event
    - Events have **duration**
  - Example
    - Schema of stock ticker stream: (Name, Price)
    - Base stream events: (IBM, 85; 9:15, 9:15), (MSFT, 27; 9:16, 9:16), (DELL, 29; 9:17, 9:17)

Cornell University

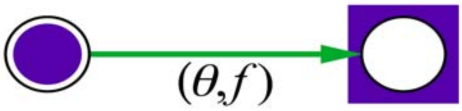
Cayuga Stream Algebra

- Compositional: Operators produce new streams from existing streams
- Translation to Nondeterministic Finite Automata
  - Edge transitions on input events
  - Automaton instances carry relevant data from matched events

Cornell University

Operators

- Relational operators (on non-temporal attributes)
  - Selection σ_θ
  - Projection π_X
  - Renaming ρ_f
  - Union
- Together these give standard pub/sub



Automaton for $\rho_f \circ \sigma_\theta(S)$

Sequence Operator

- *Sequence* operator $S_1;_\theta S_2$
- After an event from S_1 is detected, match the first event from S_2 that satisfies the condition
- Examples
  - IBM price increases by at least $1 in two consecutive sales:

$$\sigma_{p2>p1+1}(\sigma_{n1=IBM}(S_1);_{n2=IBM} S_2)$$

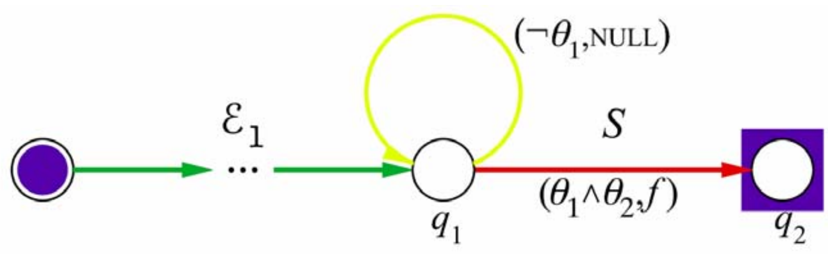
  - Find a stock whose price stays constant in two consecutive sales:

$$\sigma_{p1=p2}(S_1;_{n1=n2} S_2)$$

Sequence Operator (Contd.)

- Sequencing is a weak join on timestamps
  - Can join an event with one later in future...
  - Or with the immediate successor
    - Can be useful for queries about causal relationships

Automaton for $\rho_f \circ \sigma_{\theta_2}(\mathcal{E}_1 ;_{\theta_1} S)$

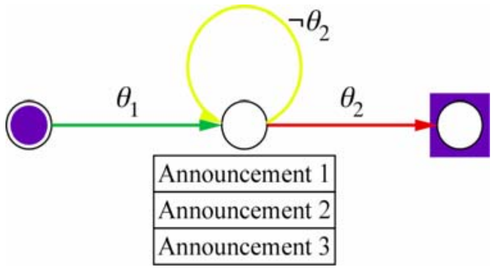


Sequence Operator: Example

- Query 1:
  - Send me the first new posting from apple.slashdot.org after a product announcement on www.apple.com.

Sequence Operator (Contd.)

- Automaton edges search for matches.
  - θ1: www.apple.com announcement
  - θ2: apple.slashdot.org posting
- Intermediate state stores Apple announcements
  - Waits to pair with next available Slashdot post.

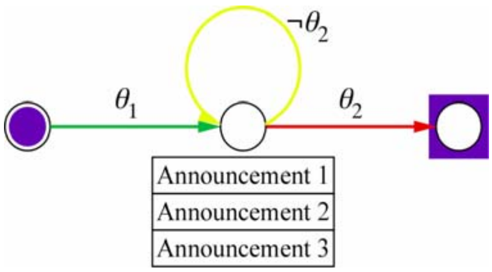


Parameterized Sequencing

- Problems with previous query
  - Assumes a quick response to Apple announcements
  - There may be several announcements (i.e., MacWorld Expo)
- Want Slashdot post to refer to right product
  - Post has link to announcement as a parameter
- Query 2:
  - Once a new product announcement appears on www.apple.com, send me the first posting from apple.slashdot.org that links to *this* announcement.

Parameterized Sequencing (Contd.)

- Intermediate information is already there
  - Each announcement is an automaton instance
- Just change edge filters to leverage information
  - θ_1: www.apple.com announcement
  - θ_2: apple.slashdot.org posting linking to an instance



Cornell University

Iteration Operator

- *Iteration* operator (similar to Kleene-+)

$$\mu_{\mathfrak{F},\theta}(S_1, S_2)$$

- Intuitively:

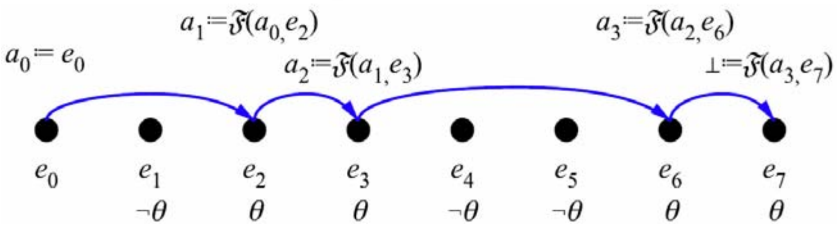
$$\mathfrak{F}(S_1 ;_\theta S_2) \cup \mathfrak{F}(\mathfrak{F}(S_1 ;_\theta S_2) ;_\theta S_2) \cup \cdots$$

Cornell University

Iteration Example

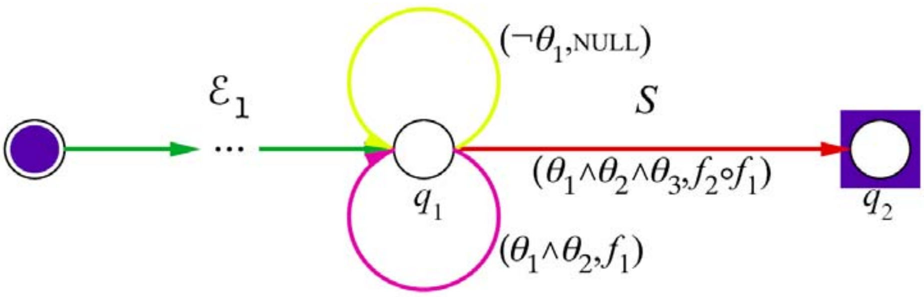
- IBM stock price monotonically increases

$$\mu_{\underbrace{\sigma_{p2>p2.last}}_{\mathfrak{F}},\underbrace{n2=IBM}_{\theta}}(\sigma_{n1=IBM}(S_1), S_2)$$



Name	IBM	MSFT	IBM	IBM	DELL	MSFT	IBM	IBM
Price	85	27	85.5	85.7	29	27.4	85.9	85.6

Automaton for Iteration Operator



Automaton for $\rho_{f_2} \circ \sigma_{\theta_3}(\mu_{\rho_{f_1} \circ \sigma_{\theta_2}, \theta_1}(\mathcal{E}_1, S))$

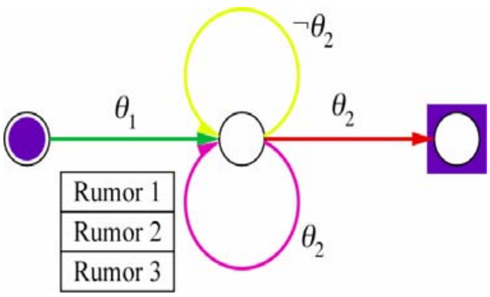
Iteration: Another Example

- Following the spread of crazy Apple rumors...

- Query 3:
  - Send me a sequence of Apple blog postings, in which the first posting is a rumor about an upcoming Apple product announcement, and each later posting is a reference (i.e., contains a direct quote from or a hypertext link to) to the previous.

Cornell University

Implementing Iteration



- Similar to parameters sequencing
  - θ_1: Initial Apple rumor
  - θ_2: Rumor that references the previous one
- Purple edge is a *rebind edge*
  - Updates instance information with latest rumor

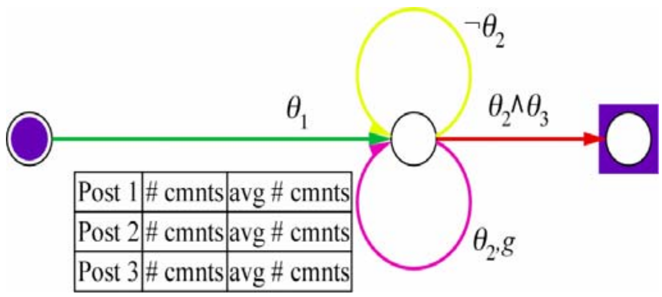
Cornell University

Aggregation

- Recall: Iteration also allows for aggregation.
  - Iterate over all posts of this type
  - Keep a running aggregate of some post attribute
    - e.g. current number of comments, average word count, etc...,
  - Implemented like normal aggregates
    - Need initializer, iterator, finalizer
- Query 4:
  - Send me an product review from apple.slashdot.org once it receives an above average number of user comments.

Cornell University

Implementing Aggregation



- Rebind edge performs the aggregation
  - *g* is attached to rebind edge to update values
- Note outgoing edge different from rebind edge
  - θ_3: Above average number of comments

Cornell University

Other Features

- Resubscription
  - Ability for one query to subscribe to the output of another (as a stream)
  - Significantly more expressive
- Extensibility
  - incorporate user-defined datatypes, data mining algorithms, predicates, aggregation functions, ...

Cornell University

Example

- Notify me when
  1. for any stock, there is a a very large trade (volume > 10K);
  2. followed by a monotonic decrease in price for at least 10 minutes;
  3. the next quote on the same stock after this monotonic sequence is 5% above the previously seen (bottom) price.
- Intuition: Large sale, followed by price drop, followed by sudden upwards move

Cornell University

Example

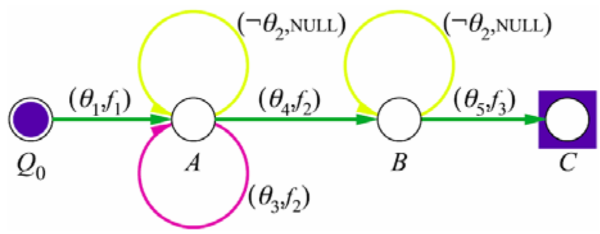
- Algebra expression: $\sigma_{\theta_5}(\sigma_{\theta_4}(\mu_{\sigma_{\theta_3},\theta_2}(S_1, S_2)) ;_{\theta_2} S_3)$

$$
\begin{aligned}
S_1 &\equiv \rho_{f_1} \circ \pi_{\texttt{name,price}} \circ \sigma_{\theta_1}(S) \\
S_2 &\equiv \rho_{f_2} \circ \pi_{\texttt{name,price}}(S) \\
S_3 &\equiv \rho_{f_3} \circ \pi_{\texttt{name,price}}(S)
\end{aligned}
$$

$$
\begin{aligned}
\theta_1 &\equiv \texttt{vol} > 10,000 \\
\theta_2 &\equiv \texttt{company} = \texttt{company.last} \\
\theta_3 &\equiv \theta_2 \wedge \texttt{minP} < \texttt{minP.last} \\
\theta_4 &\equiv \theta_3 \wedge \text{DUR} \geq 10\,\text{min} \\
\theta_5 &\equiv \theta_2 \wedge \texttt{price} > 1.05\,\texttt{minP} \\
f_1 &\equiv (\texttt{name}, \texttt{price}) \mapsto (\texttt{company}, \texttt{maxP}) \\
f_2 &\equiv (\texttt{name}, \texttt{price}) \mapsto (\texttt{company}, \texttt{minP}) \\
f_3 &\equiv (\texttt{name}, \texttt{price}) \mapsto (\texttt{company}, \texttt{finalP})
\end{aligned}
$$

Cornell University

Example



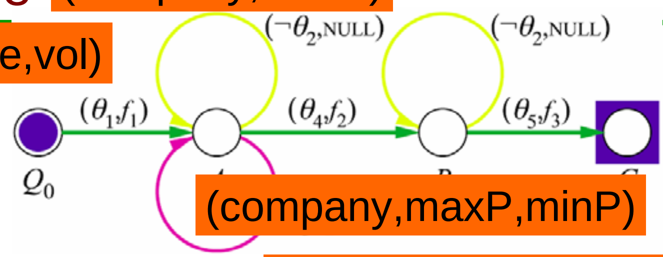
$$
\begin{aligned}
\theta_1 &\equiv \texttt{vol} > 10,000 \\
\theta_2 &\equiv \texttt{company} = \texttt{company.last} \\
\theta_3 &\equiv \theta_2 \wedge \texttt{minP} < \texttt{minP.last} \\
\theta_4 &\equiv \theta_3 \wedge \text{DUR} \geq 10\,\text{min} \\
\theta_5 &\equiv \theta_2 \wedge \texttt{price} > 1.05\,\texttt{minP} \\
f_1 &\equiv (\texttt{name}, \texttt{price}) \mapsto (\texttt{company}, \texttt{maxP}) \\
f_2 &\equiv (\texttt{name}, \texttt{price}) \mapsto (\texttt{company}, \texttt{minP}) \\
f_3 &\equiv (\texttt{name}, \texttt{price}) \mapsto (\texttt{company}, \texttt{finalP})
\end{aligned}
$$

Cornell University

Example

(name,price,vol)

(company,maxP)



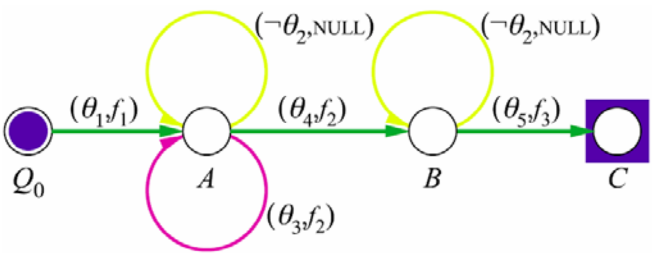
(company,maxP,minP)

(company,maxP,minP,finalP)

$$\theta_1 \equiv \texttt{vol} > 10,000$$
$$\theta_2 \equiv \texttt{company} = \texttt{company.last}$$
$$\theta_3 \equiv \theta_2 \wedge \texttt{minP} < \texttt{minP.last}$$
$$\theta_4 \equiv \theta_3 \wedge \text{DUR} \geq 10\,\text{min}$$
$$\theta_5 \equiv \theta_2 \wedge \texttt{price} > 1.05\,\texttt{minP}$$
$$f_1 \equiv (\texttt{name}, \texttt{price}) \mapsto (\texttt{company}, \texttt{maxP})$$
$$f_2 \equiv (\texttt{name}, \texttt{price}) \mapsto (\texttt{company}, \texttt{minP})$$
$$f_3 \equiv (\texttt{name}, \texttt{price}) \mapsto (\texttt{company}, \texttt{finalP})$$

Cornell University

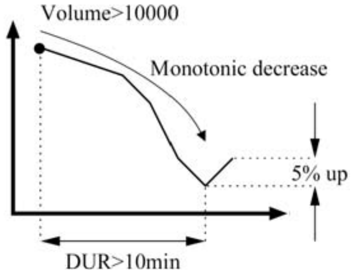
Example



Input event (name, price, vol)	Instances at state A (company, maxP, minP)	Instances at state B (company, maxP, minP)	Instances at state C (company, maxP, minP, finalP)
e_1 : ⟨IBM, 90, 15000; 9:10; 9:10⟩	I_1 = ⟨IBM, 90, NULL; 9:10; 9:10⟩		
e_2 : ⟨IBM, 85, 7000; 9:15; 9:15⟩	I_1 = ⟨IBM, 90, 85; 9:10; 9:15⟩		
e_3 : ⟨Dell, 40, 11000; 9:17; 9:17⟩	I_1 = ⟨IBM, 90, 85; 9:10; 9:15⟩ I_2 = ⟨Dell, 40, NULL; 9:17; 9:17⟩		
e_4 : ⟨IBM, 81, 8000; 9:21; 9:21⟩	I_1 = ⟨IBM, 90, 81; 9:10; 9:21⟩ I_2 = ⟨Dell, 40, NULL; 9:17; 9:17⟩	I_3 = ⟨IBM, 90, 81; 9:10; 9:21⟩	
e_5 : ⟨MSFT, 25, 6000; 9:23; 9:23⟩	I_1 = ⟨IBM, 90, 81; 9:10; 9:21⟩ I_2 = ⟨Dell, 40, NULL; 9:17; 9:17⟩	I_3 = ⟨IBM, 90, 81; 9:10; 9:21⟩	
e_6 : ⟨IBM, 91, 9000; 9:24; 9:24⟩	I_2 = ⟨Dell, 40, NULL; 9:17; 9:17⟩		I_3 = ⟨IBM, 90, 81, 91; 9:10; 9:24⟩

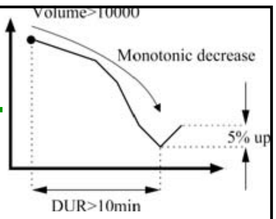
Cornell University

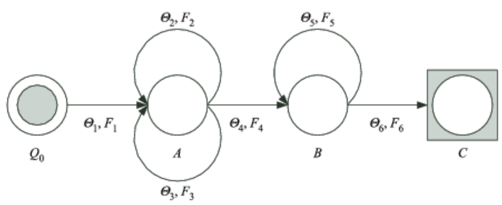
Cayuga Query Language




```
SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
  FILTER{Price > 1.05*MinPrice}(
    FILTER{DUR > 10min}(
     (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
     FROM FILTER{Volume > 10000}(Stock))
       FOLD{$2.Name = $.Name, $2.Price < $.Price}
     Stock)
    NEXT{$2.Name = $1.Name}
    Stock)
```

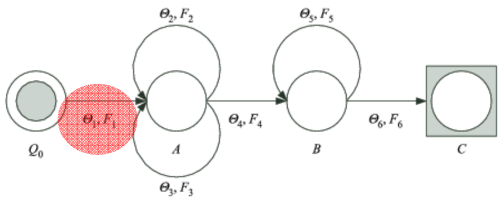
Cayuga Automata






```
SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
  FILTER{Price > 1.05*MinPrice}(
    FILTER{DUR > 10min}(
     (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
     FROM FILTER{Volume > 10000}(Stock))
       FOLD{$2.Name = $.Name, $2.Price < $.Price}
     Stock)
    NEXT{$2.Name = $1.Name}
    Stock)
```

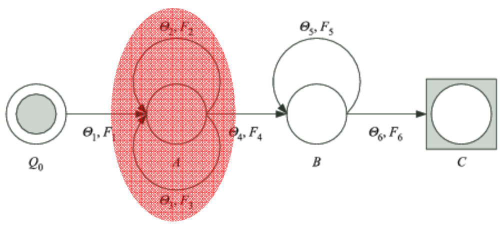
Cayuga Automata




```
SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
  FILTER{Price > 1.05*MinPrice}(
    FILTER{DUR > 10min}(
     (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
     FROM FILTER{Volume > 10000}(Stock))
       FOLD{$2.Name = $.Name, $2.Price < $.Price}
     Stock)
    NEXT{$2.Name = $1.Name}
    Stock)
```

Cornell University

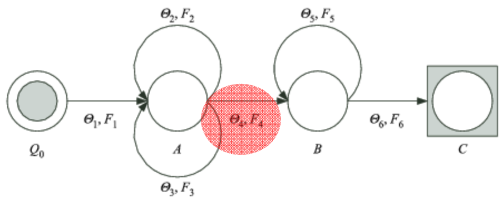
Cayuga Automata




```
SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
  FILTER{Price > 1.05*MinPrice}(
    FILTER{DUR > 10min}(
     (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
     FROM FILTER{Volume > 10000}(Stock))
       FOLD{$2.Name = $.Name, $2.Price < $.Price}
     Stock)
    NEXT{$2.Name = $1.Name}
    Stock)
```

Cornell University

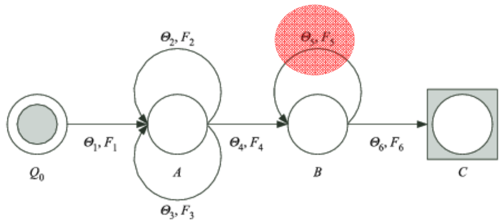
Cayuga Automata



SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
  FILTER{Price > 1.05*MinPrice}(
    FILTER{DUR > 10min}(
      (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
      FROM FILTER{Volume > 10000}(Stock))
        FOLD{$2.Name = $.Name, $2.Price < $.Price}
      Stock)
    NEXT{$2.Name = $1.Name}
    Stock)

Cornell University

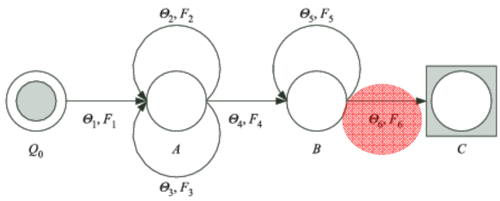
Cayuga Automata



SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
  FILTER{Price > 1.05*MinPrice}(
    FILTER{DUR > 10min}(
      (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
      FROM FILTER{Volume > 10000}(Stock))
        FOLD{$2.Name = $.Name, $2.Price < $.Price}
      Stock)
    NEXT{$2.Name = $1.Name}
    Stock)

Cornell University

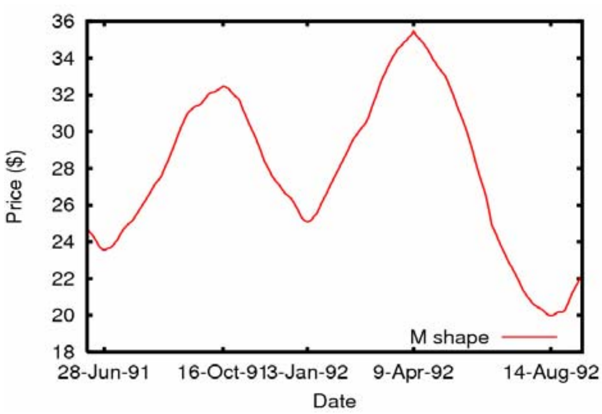
Cayuga Automata



SELECT Name, MaxPrice, MinPrice, Price AS FinalPrice
FROM
  FILTER{Price > 1.05*MinPrice}(
    FILTER{DUR > 10min}(
      (SELECT Name, Price_1 AS MaxPrice, Price AS MinPrice
      FROM FILTER{Volume > 10000}(Stock))
        FOLD{$2.Name = $.Name, $2.Price < $.Price}
      Stock)
    NEXT{$2.Name = $1.Name}
    Stock)

Example: Double-Top

- Double-Top query pattern



Cayuga Resubscription (No Iteration)

- Compute stream of local extrema: $\sigma_{\theta_1}(\sigma_{\theta_2}(S_1 \underset{\theta_3}{;} S_2) \underset{\theta_4}{;} S_3)$
- Union them, then search for actual pattern:

$$\sigma_{\theta_1}(\sigma_{\theta_2}(\sigma_{\theta_3}(\sigma_{\theta_4}(E_1 \underset{\theta_5}{;} E_2) \underset{\theta_6}{;} E_3) \underset{\theta_7}{;} E_4) \underset{\theta_8}{;} E_5)$$

$$\begin{aligned}
\theta_1 &\equiv (E_5.\mathtt{price} \leq E_1.price) \\
\theta_2 &\equiv (0.9E_2.\mathtt{price} \leq E_4.\mathtt{price} \leq 1.1E_2.\mathtt{price}) \\
\theta_3 &\equiv (0.9E_1.\mathtt{price} \leq E_3.\mathtt{price} \leq 1.1E_1.\mathtt{price}) \\
\theta_4 &\equiv (E_2.\mathtt{price} \geq 1.2 * E_1.\mathtt{price}) \\
\theta_5 &\equiv (E_2.\mathtt{name} = E_1.\mathtt{name}) \\
\theta_6 &\equiv (E_3.\mathtt{name} = E_1.\mathtt{name}) \\
\theta_7 &\equiv (E_4.\mathtt{name} = E_1.\mathtt{name}) \\
\theta_8 &\equiv (E_5.\mathtt{name} = E_1.\mathtt{name})
\end{aligned}$$

Double-Top Query: Cayuga

```
SELECT Name, PriceA, PriceB, PriceC, PriceD, Price_1 AS PriceE, Price AS PriceF
FROM FILTER {Price >= Price_1 AND Price <= PriceA}
        (FILTER{Price <= 1.1*PriceB} (
        SELECT Name, PriceA, PriceB, PriceC, Price_1 AS PriceD, Price
        FROM
        FILTER{Price >= 0.9*PriceB} (
        SELECT Name, PriceA, PriceB, Price_1 AS PriceC, Price
        FROM
        FILTER{Price >= 0.9*PriceA AND Price <= 1.1*PriceA} (
        SELECT Name, PriceA, Price_1 AS PriceB, Price
        FROM
        FILTER{Price >= 1.2*PriceA} (
        SELECT Name, Price_1 AS PriceA, Price
        FROM
        FILTER {Price < Price_1}
        (SELECT Name, Price FROM Stock NEXT {$1.Name=$2.Name} Stock)
        FOLD {$1.Name = $2.Name, $2.Price >= $.Price,} Stock)
        FOLD {$1.Name = $2.Name, $2.Price <= $.Price,} Stock)
        FOLD {$1.Name = $2.Name, $2.Price >= $.Price,} Stock)
        FOLD {$1.Name = $2.Name, $2.Price <= $.Price,} Stock)
                NEXT {$1.Name = $2.Name}
  Stock)
PUBLISH MShapeStock
```

Double-Top Query: CQL

vquery : Rstream (Select S.time, S.name, S.price, (S.price - P.price)
From Stock [Now] as S, Stock [Partition By P.name Rows 2] as P Where S.name = P.name and S.time > P.time);
vtable : register stream StockDiff (time integer, name integer, price float, pdiff float);

vquery : Rstream (Select P.time, P.name, P.price, P.pdiff
From StockDiff [Now] as S, StockDiff [Partition By P.name Rows 2] as P Where S.name = P.name and (S.pdiff * P.pdiff) < 0.0);
vtable : register stream Extrema (time integer, name integer, price float, pdiff float);

vquery : Select name, count(*) from Extrema Group By name;
vtable : register relation ExtremaCounter (name integer, seqNo integer);

vquery : Rstream (Select E.name, E.price, E.pdiff, C.seqNo, C.seqNo – 1
From Extrema [Now] as E, ExtremaCounter as C Where E.name = C.name);
vtable : register stream ExtremaSeq (name integer, price float, pdiff float, seq integer, prevSeq integer);

vquery : Select name, price, seq from ExtremaSeq Where pdiff < 0.0;
vtable : register relation stateA (name integer, price float, seq integer);

vquery : Rstream (Select E.name, E.price, A.price, E.seq
From ExtremaSeq [Now] as E, stateA as A Where E.name = A.name and E.prevSeq = A.seq and E.price > (A.price * 1.2));
vtable : register relation stateB (name integer, bprice float, aprice float, seq integer);

vquery : Rstream (Select E.name, E.price, B.bprice, B.aprice, E.seq From ExtremaSeq [Now] as E, stateB as B
Where E.name = B.name and E.prevSeq = B.seq and E.price > (B.aprice * 0.9) and E.price < (B.aprice * 1.1));
vtable : register relation stateC(name integer, cprice float, bprice float, aprice float, seq integer);

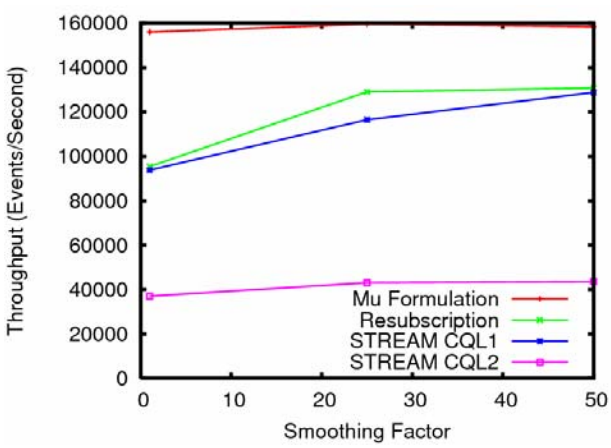
vquery : Rstream (Select E.name, E.price, C.cprice, C.bprice, C.aprice, E.seq from ExtremaSeq [Now] as E, stateC as C
Where E.name = C.name and E.prevSeq = C.seq and E.price > (C.bprice * 0.9) and E.price < (C.bprice * 1.1));
vtable : register relation stateD (name integer, dprice float, cprice float, bprice float, aprice float, seq integer);

query : Rstream (Select E.name, E.price, D.dprice, D.cprice, D.bprice, D.aprice from ExtremaSeq [Now] as E, stateD as D
Where E.name = D.name and E.prevSeq = D.seq and E.price <= D.aprice);

Cornell University

Example: Double-Top

- Real stock data (24 companies, 112,635 events)



Cornell University